Universidad Nacional Autónoma de México

Facultad de Ciencias

# Un acercamiento a la gramática vía Teoría de categorías

**T E S I S**

que para obtener el título de:

Licenciado en Matemáticas

P R E S E N T A :

Fernando Herrera Ferreyra

TUTOR

Dr. Omar Antolín Camerena

Ciudad Universitaria, CDMX, 2024

# Índice general

# 1 Introducción

hola

# 2 Gramáticas regulares

En el presente capítulo comenzaremos con la definición de una categoría libre junto a los lenguajes que producen. Posteriormente, probaremos que son equivalentes a los lenguajes regulares, aquellos producidos por un autómata finito, y finalizaremos mostrando la implementación en `Python`.

## 2.1. Categorías libres

**¿incluir la definición de categoría?**

**Definición 1.** Una **signatura simple** $G$ (o gráfica dirigida) es una conjunto de vértices $G_0$ y uno de aristas $G_1$ tal que cada arista tiene asociado dos vértices, un dominio y un codominio

$$G_0 \xleftarrow{\texttt{dom}} G_1 \xrightarrow{\texttt{cod}} G_0$$

Dadas $G, \Gamma$ signaturas simples, un **homomorfismo de signaturas** $\varphi : G \to \Gamma$ es un par de funciones $\varphi_0 : G_0 \to \Gamma_0$ y $\varphi_1 : G_1 \to \Gamma_1$ tal que el siguiente diagrama conmuta:

$$\begin{array}{ccccc} G_0 & \xleftarrow[\texttt{cod}]{} & G_1 & \xrightarrow{\texttt{dom}} & G_0 \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0 & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_0 \end{array}$$

Observemos que si consideramos dos homomorfismos de signaturas $\varphi = (\varphi_0, \varphi 1) : G \to \Gamma$ y $\psi = (\psi_0, \psi_1) : \Gamma \to \Phi$ podemos tomar su composición entrada a entrada, es decir, $\psi \circ \varphi = (\psi_0 \circ \varphi_0, \psi_1 \circ \varphi_1)$, la cual resulta nuevamente un homomorfismo de signaturas, pues el siguiente diagrama

$$\begin{array}{ccccc} G_0 & \xleftarrow[\texttt{cod}]{} & G_1 & \xrightarrow{\texttt{dom}} & G_1 \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0 & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_1 \\ \downarrow{\scriptstyle \psi_0} & & \downarrow{\scriptstyle \psi_1} & & \downarrow{\scriptstyle \psi_0} \\ \Phi_0 & \xleftarrow[\texttt{cod}]{} & \Phi_1 & \xrightarrow{\texttt{dom}} & \Phi_1 \end{array}$$

conmuta ya que el superior y el inferior lo hacen.
Notemos, además, que la composición anterior es asociativa, pues lo es en cada entrada y que la identidad $Id_\Gamma = (Id_{\Gamma_0}, Id_{\Gamma_1})$ satisface que $Id_\Gamma \circ \varphi = \varphi$ y $\psi \circ Id_\Gamma = \psi$. Por lo tanto, tenemos una categoría cuyos objetos son las signaturas simples y sus flechas son los homomorfismos, la cual denotaremos **Graph**.

Por otro lado, podemos considerar la categoría **Cat** cuyos objetos son categorías y las flechas son funtores entre ellas. Con esto, definimos el siguiente funtor:

$$\mathbf{U} : \mathbf{Cat} \to \mathbf{Graph}$$

tal que para cada categoría $\mathcal{C}$, $\mathbf{U}\mathcal{C}$ es la gráfica dirigida cuyos vértices son los objetos, $Ob(\mathcal{C})$, y los aristas corresponden a los morfismos, $Mor(\mathcal{C})$. Además, notemos que cada funtor $f : \mathcal{C} \to \mathcal{C}'$ induce a un homomorfismo de gráficas $\mathbf{Uf} : \mathbf{U}(\mathcal{C}) \to \mathbf{U}(\mathcal{C}')$. Puesto que el funtor $\mathbf{U}$ lo único que hace es olvidar la composición de morfismos en una categoría para obtener la gráfica asociada, decimos que es un funtor olvidadizo.
Ahora, nuestro objetivo es definir un funtor

$$\mathbf{F} : \mathbf{Graph} \to \mathbf{Cat}$$

.
Dada una gráfica dirigida $G$, los objetos de $\mathbf{FG}$ serán los vértices de $G$, y sus morfismos serán de las siguientes dos formas:

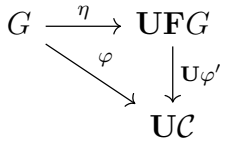
- Para cada $v$ vértice de $G$, tenemos un morfismo $Id_v : v \to v$; y
- para cualquier $n \in \mathbb{N}$ y para cualesquiera $e_1, ..., e_n$ aristas de $G$ tales que $\mathtt{cod}(\mathtt{e_i}) = \mathtt{dom}(\mathtt{e_{i+1}})$ con $i \in \{1, ..., n-1\}$ tenemos un morfismo $e_1 \dots e_n : \mathtt{dom}(\mathtt{e_1}) \to \mathtt{cod}(\mathtt{e_n})$.

Es decir, los morfismos corresponden a las identidades y a todos los posibles caminos dentro de la gráfica. La composición de dos caminos está dada por la concatenación, mientras que componer con la identidad actúa de la manera esperada.
Como la concatenación es siempre asociativa, claramente tenemos que $\mathbf{F}(\mathbf{G})$ es una categoría.
Ahora, tomemos $\varphi : G \to G'$ un homomorfismo de gráficas, definimos el funtor $\mathbf{F}\varphi : \mathbf{F}(\mathbf{G}) \to \mathbf{F}(\mathbf{G}')$ de la siguiente manera:

- Para cada $v$ objeto en $\mathbf{F}(G)$, $(\mathbf{F}\varphi)(v) = \varphi_0(v)$;
- Para cada $I_v$ morfismo identidad de $\mathbf{F}(G)$, $(\mathbf{F}\varphi)(I_v) = I_{\varphi_0(v)}$; y
- Para cada $E : e_1 \dots e_n$ morfismo camino de $\mathbf{F}(G)$, $(\mathbf{F}\varphi)(e_1 \dots e_n) = \varphi(e_1) \dots \varphi(e_n)$

Hagamos notar que el último inciso está bien definido pues $\varphi$ es un homomorfismo de gráficas.
Con lo anterior, estamos en condiciones de probar nuestra primera proposición:

**Proposición 1.** *Sea $G$ una gráfica dirigida. Entonces existe un morfismo de gráficas $\eta : G \to \mathbf{UF}G$ tal que para cualquier categoría $\mathcal{C}$ y un homomorfismo de gráficas $\varphi : G \to \mathbf{U}\mathcal{C}$ existe un único funtor $\varphi' : \mathbf{F}G \to \mathcal{C}$ que hace conmutar el siguiente diagrama:*




*Demostración.* Definimos $\eta : G \to \mathbf{UF}G$ como $\eta_1$ la identidad en vértices y dado un arista $e : v_1 \to v_2$ en $G$, entonces $\bar{e}$ es un morfismo en $\mathbf{F}G$, y así $\eta(e) = e : v_1 \to v_2$ un arista en $\mathbf{UF}G$. Claramente es un morfismo de gráficas. **Pendiente.** □

Observemos que el teorema nos indica que dada una gráfica $G$, el funtor $\mathbf{F}$ construye la categoría libre generada por $G$, la cual denotaremos como $\mathbf{C}(G)$

**Proposición 2.** *El funtor libre* $\mathbf{F} : \mathbf{Graph} \to \mathbf{Cat}$ *es adjunto izquierdo del funtor olvidadizo* $\mathbf{U} : \mathbf{Cat} \to \mathbf{Graph}$

*Demostración.* **Pendiente** □

## 2.2. Lenguaje en una categoría libre

Nuestro objetivo ahora será definir el lenguaje generado por una signatura simple (o gráfica dirigia) y posteriormente mostrar que son exactamente los lenguajes regulares.
Sea $V$ un vocabulario (o léxico) no vacío y $G$ una signatura simple. A cada arista de $G$ le asociaremos una palabra en $V$, mediante una función $L : G_0 \to V$, o equivalentemente, un homomorfismo de gráficas $L : G \to V$ donde $V$ es la gráfica con un vértice y cada palabra es un arista. Observemos que podemos extender $L$ de la siguiente forma: para cada morfismo $f = f_n \circ \cdots \circ f_1$ en $\mathbf{C}(G)$, $L^*(f) = L(f_1) \cdots L(f_n) \in V^*$ y $L^*(Id_v) = \varepsilon$ para cualquier $v \in G_0$.

**Definición 2.** Una **gramática simple** es un signatura finita simple $G$ junto a un homomorfismo de gráficas $L : G \to V$ donde $V$ es un conjunto de palabras, el vocabulario, y símbolos distinguidos $s_0, s_1 \in G_0$, llamados el símbolo inicial y el terminal. Así, tenemos el siguiente diagrama:

$$\begin{array}{ccccc} G_0 & \xleftarrow[\texttt{dom}]{} & G_1 & \xrightarrow{\texttt{cod}} & G_0 \\ & & \downarrow{\scriptstyle L} & & \\ & & V & & \end{array}$$

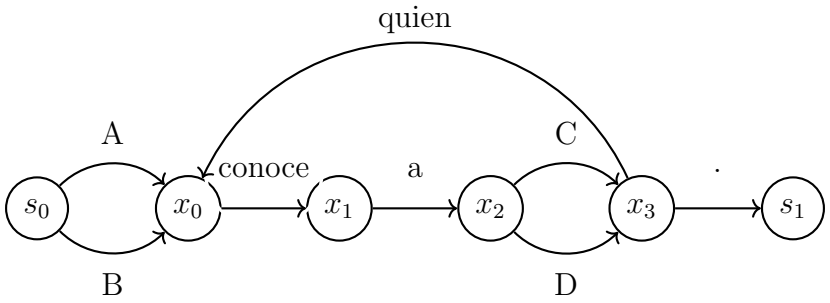
El lenguaje generado por $G$ se define como:

$$\mathcal{L}(G) = L^*(\mathbf{C}(s_0, s_1)) \subset V*$$

Un morfismo de gramáticas simples $\varphi : G \to H$ es un homomorfismo de gráficas tal que el siguiente diagrama conmuta:

$$\begin{array}{ccc} G & \xrightarrow{\varphi} & H \\ & {\scriptstyle L_G} \searrow \quad \swarrow {\scriptstyle L_H} & \\ & V & \end{array}$$

**Ejemplo 1.** Consideremos la siguiente gráfica $G$:




Observemos que induce una gramática simple, y una oración "A conoce a C quien conoce a D." está en el lenguaje generado por dicha gramática.

**Definición 3.** Una gramática regular es una tupla $(N, V, P, s_0)$ donde $V$ es un vocabulario no vacío; $N$ es un conjunto de símbolos no terminales con un símbolo de inicio $s_0$; y $P$ un conjunto finito de reglas de producción, cada una de alguna de las siguientes formas:

$$A \to aB$$

$$A \to a$$

$$A \to \varepsilon$$

con$A$ y $B$ en $N$, y $a \in V$.
Una derivación es la aplicación consecutiva de reglas de producción.
Definimos el lenguaje $L$ generado por una gramática regular como $w \in V*$ está en $L$ si y sólo si existe una derivación de $s_0$ a $w$.
Decimos que un lenguaje es regular si es generado por un gramática regular.

**Ejemplo 2.** Consideremos la siguiente gramática regular con los símbolos no terminales $N = \{A, s_0\}$, el vocabulario $V = \{a, b, c\}$ y con las siguientes reglas $P$:

$$s_0 \to as_0$$
$$s_0 \to bA$$
$$A \to \varepsilon$$
$$A \to cA$$

Observemos que el lenguaje generado por dicha gramática es $L = \{a^i bc^j | i, j \in \mathbb{N}\}$

Con la definición anterior, ya estamos en condiciones para demostrar el teorema principal de este capítulo:

**Teorema 1.** *(a) Todo lenguaje regular es generado por una gramática simple.*

*(b) Todo lenguaje generado por una gramática simple es regular.*

*Demostración.* Primero probemos $(a)$.
Sea $L$ un lenguaje regular generado por la gramática $(N, V, P, s)$.
Definimos una gramática simple $G = P \rightrightarrows (V \cup \{s, \varepsilon\})$ donde para cada $p$ regla de producción:

- Si $p$ es de la forma $A \to aB$ con $A, B \in N$ y $a \in V$ entonces $p : A \to B$ en $G$ y $L(p) = a$.
- Si $p$ es de la forma $A \to a$ con $A \in N$ y $a \in V$, entonces $p : A \to s_1$ en $G$ y $L(p) = a$
- Si $p$ es de la forma $A \to \varepsilon$ con $A \in N$, entonces $p : A \to s_1$ en $G$ y $L(p) = \varepsilon$

Observación. El símbolo $\varepsilon$ de nuestro vocabulario cubrirá el papel de la cadena vacía.

Probemos que $L = L(G)$.
Sea $w \in L$. Entonces existe una derivación

$$s_0 \xrightarrow{p_1} \cdots \xrightarrow{p_n} w$$

Observemos que si $n = 1$, entonces $p_1$ es de la forma $s \to w$, así tenemos el morfismo $p_1 : s_0 \to s_1$ tal que $L(p_1) = w$. Ahora, si $n > 1$ veamos que la derivación es necesariamente de la forma:

$$s_0 \xrightarrow{p_1} aA_1 \xrightarrow{p_2} \cdots \xrightarrow{p_{n-1}} uA_{n-1} \xrightarrow{p_n} w$$

donde $A_i \in N$ con $i \in \{1, ..., n-1\}$, $a \in V$ y $u \in V^*$. Así tenemos los siguientes morfismos en $G$:

$$s_0 \xrightarrow{p_1} A_1 \xrightarrow{p_2} \cdots \xrightarrow{p_{n-1}} A_{n-1} \xrightarrow{p_n} s_1$$

Entonces el morfismo $p = p_n \circ p_{n-1} \circ \cdots \circ p_2 \circ p_1 : s_0 \to s_1$ y

$$\begin{aligned} L(p) = L(p_n \circ p_{n-1} \circ \cdots \circ p_2 \circ p_1) &= (L(p_1)L(p_2)\dots L(p_n))L(p_{n-1}) \\ &= uL(p_{n-1}) = w \end{aligned}$$

Observemos que es posible que $L(p_{n-1}) = \varepsilon$ pero en ese caso $u = u\varepsilon = w$.
Por lo tanto, $w \in L(G)$. Así, $L \subset L(G)$
Ahora, sea $w \in L(G)$. Entonces existe $f : s_0 \to s_1$ en $\mathbf{C}(G)$ tal que $L(f) = w$. Como $\mathbf{C}(G)$ es una categoría libre, entonces $f$ es un camino en $G$, así existen $p_1, \dots, p_n$ reglas de producción en $P$ tal que $f = p_n \circ ... \circ p_1$ que inducen un derivación:

$$s_0 \xrightarrow{p_1} \cdots \xrightarrow{p_n} w$$

Y, por lo tanto, $w \in L$. Así $L(G) \subset L$.
Con ello concluimos que $L = L(G)$ y terminamos la prueba del inciso (a).
Vayamos con la prueba de (b)
Sea $G$ una gramática simple con el homomorfismo $L : G \to V$ con $V$ vocabulario.
La estrategia será construir un autómata finito no determinista que admite el lenguaje $L(G)$, lo cual implica que $L(G)$ es un lenguaje regular.
**Agregar referencia**
Definimos el AFND $M$ de la siguiente manera: su conjunto de estados es $G_1$, es decir, los vértices de $G$, donde $s_0$ es el símbolo de inicio y $F = \{s_1\}$ el conjunto de estados finales, y tomemos $V$ como el conjunto de entradas. Definimos la función de transición $\delta : G_0 \times V \to 2^{G_0}$ tal que para cada $q \in G_0$ y $v \in V$,

$$\delta(q, v) = \{q' | \exists f : q \to q' \in G_1 \text{ tal que } L(f) = v\}$$

Recordemos que $\delta$ se extiende recursivamente a $V*$ de la siguiente manera:

$$\delta^*(q,\varepsilon) = \{q\}$$
$$\delta^*(q,xa) = \bigcup_{r\in\delta^*(q,a)} \delta(r,a)$$

Así, definimos el lenguaje aceptado por $M$ como:

$$L(M) = \{x \in V^* | \delta^*(s_0,x) \cap s_1 \neq \varnothing\}$$

Probemos que $L(G) = L(M)$.
Sea $w \in L(G)$. Entonces existe $f : s_0 \to s_1$ en $\mathbf{C}(G)$ tal que $L(f) = q$. Observemos que entonces existen $f_1, \ldots, f_n$ en $G_1$ tales que el siguiente diagrama conmuta:

$$a_0 = s_0 \xrightarrow{f_1} a_1 \xrightarrow{f_2} \ldots \xrightarrow{f_{n-1}} a_{n-1} \xrightarrow{f_n} s_1 = a_{n+1}$$

Así $q = L(f) = L(f_1)\cdots L(f_n)$, entonces observemos que para cada $i \in \{0,...,n\}$, $a_{i+1} \in \delta(a_i, L(f_i))$ y, por lo tanto, $s_1 = a_{n+1} \in \delta^*(a_0, L(f)) = \delta^*(s_1, w)$. Como $\{s_1\}$ es el conjunto terminal, concluimos que $w \in L(M)$. Por lo tanto, $L(G) \subset L(M)$.
Sea $w \in L(M)$
Entonces $s_1 \in \delta^*(s_0, w)$.
Observemos que si $w = \varepsilon$, entonces $s_1 \in \delta(s_0, \varepsilon)$, así hay un morfismo $f : s_0 \to s_1$ tal que $L^*(f) = \varepsilon$, por la definición de $L^*$ esto implica que $f = Id_{s_0}$, así $s_0 = s_1$ y $w \in L(G)$.
Por otro lado, si $w$ no es la cadena vacía, entonces hay $w_1, \ldots, w_n \in V$ tales que $w = w_1 \cdots w_n$ y estados $q_1, \ldots, q_{n-1}$ que satisfacen:

$$q_1 \in \delta(s_0, w_1) \qquad q_{i+1} \in \delta(q_i, w_i) \qquad s_0 \in \delta(q_{n-1}, w_n)$$

para $i \in \{1, ..., n-2\}$. Esto significa que hay $f_1, \ldots, f_n$ en $G_1$ tales que

$$s_0 \xrightarrow{f_1} q_1 \xrightarrow{f_2} \ldots \xrightarrow{f_{n-1}} q_{n-1} \xrightarrow{f_n} s_1$$

y $L(f_i) = w_i$ con $i \in \{1, \ldots n\}$. Por lo tanto, $f_n \circ ... \circ f_1 : s_0 \to s_1 \in \mathbf{C}(G)$ y

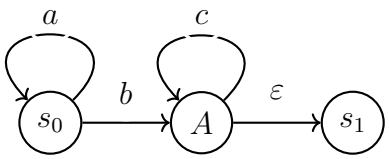
$$L^*(f_n \circ ... \circ f_1) = L(f_1)\cdots L(f_n) = w_1 \cdots w_n = w$$

Por lo tanto, $w \in L(G)$ y así $L(M) \subset L(G)$.
Por ello, $L(G) = L(M)$ como queríamos demostrar. □

A partir de ahora dejaremos de usar el término gramáticas simples y, en virtud del teorema anterior, las llamaremos gramáticas regulares. Denotaremos como $\mathbf{Reg_V}$ a la categoría cuyos objetos son gramáticas regulares y los morfismos son los homomorfismos entre ellas.

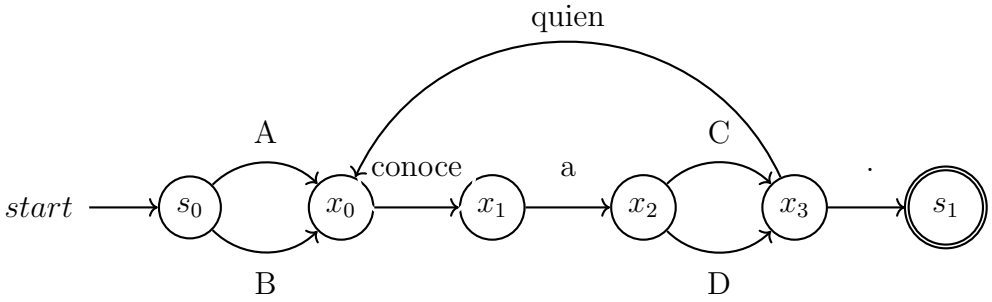
Para ilustrar las pruebas anteriores, veamos los siguientes ejemplos:

**Ejemplo 3.** Consideremos la gramática regular del segundo ejemplo, observemos que su lenguaje es generado por la gramática de la siguiente gráfica:




Vale la pena notar que esta gráfica induce inmediatamente un autómata finito con épsilon transiciones.

**Ejemplo 4.** Consideremos la gramática regular del primer ejemplo, es claro que induce el siguiente AFND:




Un resultado bien conocido sobre lenguajes regulares es que son cerrados respecto a uniones e intersección finitas, la prueba usual es sencilla utilizando que los lenguajes regulares son exactamente los generados por expresiones regulares, sin embargo, veamos una prueba en términos categóricos.

**Proposición 3.** *Los lenguajes regulares son cerrados bajo intersecciones y uniones finitas.*

*Demostración.* Sean $G$ y $G'$ gramáticas regulares con símbolos iniciales $s_0$, $s'_0$ y símbolos finales $s_1$, $s'_1$ respectivamente.
Definimos la siguiente gramática regular $G \cap G' : H_0 \subset G_1 \times G'_1 \rightrightarrows H_1 \subset G_0 \times G'_0$ de la siguiente forma: $(q_0, q'_0)$ y $(q_1, q'_1)$ son el símbolo inicial y final respectivamente. Además, existe un arista entre $(a, b)$ y $(a', b')$ en $G \cap G'$ si y sólo si existen $a \xrightarrow{f} b$ en $G$ y $a' \xrightarrow{f'} b'$ en $G'$ tales que $L_G(f) = L_{G'}(f')$, denotaremos a dicho arista como $(f, f')$ y con ello definimos $L((f, f')) = L_G(f)$.
Afirmación. $L(G \cap G') = L(G) \cap L(G')$.
Tenemos que $w \in L(G \cap G')$ si y sólo si existe $(f, f') : (s_0, s'_0) \to (s_1, s'_1) \in \mathbf{C}(G \cap G')$ tal que $L^*(f, f') = w$, si y sólo existen $(f_1, f'_1), \cdots (f_n, f'_n)$ aristas de $G \cap G_1$ tales que el siguiente diagrama conmuta

$$(s_0, s'_0) \xrightarrow{(f_1.f'_1)} \ldots \xrightarrow{(f_n, f'_n)} (s_1, s'_1)$$

y $w = L(f_1, f_1') \cdots L(f_n, f_n')$, pero esto ocurre si y sólo existen $f_1, ..., f_n$ aristas en $G$ y $f_1', ..., f_n'$ aristas en $G'$ tales que los siguientes diagramas conmutan:


y $w = L(f_1) \cdots L(f_1) = L(f_1') \cdots L(f_n')$, lo cual sucede si y sólo existen $f : s_0 \to s_1$ en $\mathbf{C}(G)$ y $f' : s_0' \to s_1'$ en $\mathbf{C}(G')$ tales que $L^*(f) = w = L^*(f')$
Y esto se da si y sólo si $w \in L(G)$ y $w \in L(G')$. Por lo tanto $L(G \cap G') = L(G) \cap L(G')$.
Ahora, para probar el resultado en el caso de la unión definimos la gramática regular $G \cup G' : G_0 + G_0' \rightrightarrows G_1 + G_1'$ donde el símbolo + representa la unión disjunta y además haremos la identificación $s_0 = s_0'$ y $s_1 = s_1'$.
Afirmación. $L(G \cup G') = L(G) \cup L(G')$.
Notemos que $w \in L(G \cup G')$ si y sólo si existe $f : s_0 \to s_1$ en $\mathbf{C}(G \cup G')$ tal que $L^*(f) = w$.
Procediendo de manera análoga a la prueba anterior, veremos que esto ocurre si y sólo si existe $f : s_0 \to s_1$ en $\mathbf{C}(G)$ o $f : s_0' \to s_1'$ en $\mathbf{C}(G')$ tal que $L^*(f) = w$, pero esto se da cuando $w$ está en $L(G)$ o en $L(G')$.
Por lo tanto, $L(G \cup G') = L(G) \cup L(G')$.
Entonces, $L(G) \cap L(G')$ y $L(G) \cup L(G')$ son lenguajes regulares y, por inducción, es inmediato ver que esto es cierto para uniones e intersecciones finitas. □

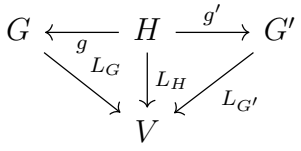
Observemos que otra forma de interpretar lo anterior, es que dado dos objetos $G, G'$ en $\mathbf{Reg_V}$ encontramos dos objetos $G \cap G'$ y $G \cup G'$ en $\mathbf{Reg_V}$. La siguiente proposición nos dirá la manera en que se relacionan en la categoría.

**Proposición 4.** $\mathbf{Reg_V}$ *tiene productos y coproductos finitos. Más aún, si* $G, G'$ *son objetos de* $\mathbf{Reg_V}$*, entonces* $G \cap G'$ *y* $G \cup G'$ *son su producto y coproducto respectivamente.* ***Probablemente debería cambiar la notación del coproducto***

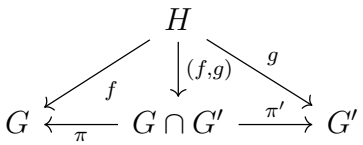
*Demostración.* Sean $G$ y $G'$ en $\mathbf{Reg_V}$.
Primero veamos que $G \cap G'$ es el producto.
Sea $H$ en $\mathbf{Reg_V}$ junto con $f : H \to G$ y $g : H \to G'$ morfismos de gramáticas regulares.
Vamos a definir el morfismo de gramáticas regulares $(f, g) : H \to G \cap G'$ de la siguiente manera:

- Para cada vértice $v \in H_0$, $(f, g)_0(v) = (f_0(v), g_0(v))$
- Para arista $h \in H_0$, $(f, g)_1(h) = (f_1(h), g_1(h))$

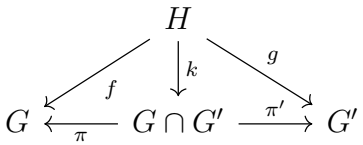
Veamos que es un morfismo de gramáticas regulares. Sea $h : v_1 \to v_2$ un arista de $H$. Como $f, g$ son morfismos de gramáticas regulares, entonces $f(h) : f(v_1) \to f(v_2)$ y $g(h) : g(v_1) \to g(v_2)$ son aristas $G$ y $G'$ respectivamente y, además, $L_G(f(h)) = L_H(h) = L_{G'}(g(h))$ pues el siguiente diagrama conmuta:




Así $(f,g)(h) : (f,g)(v_1) \to (f,g)(v_2)$ está en $G_1 \cap G_2$ y $L_H(h))L_{G\cap G'}((f,g)(h))$como queríamos ver.
Ahora, vamos a definir las proyecciónes $\pi : G \cap G' \to G$ y $\pi' : G \cap G' \to G'$ como las inducidas por las proyecciones $G \times G' \to G$ y $G \times G' \to G$, así como definimos $(f,g)$ entrada a entrada, tenemos que el siguiente diagrama conmuta:



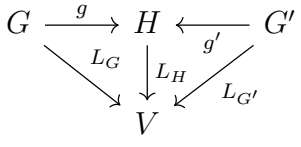

Supongamos entonces existe un morfismo de gramáticas regulares $k : H \to G \cap G'$ tal que el siguiente diagrama también conmuta:




Observemos que como $\pi$ y $\pi'$ las tomamos como las inducidas por las proyecciones del producto en aristas y vértices, entonces por unicidad en **Set** tenemos que $k_0 = (f,g)_0$ y $k_1 = (f,g)_1$ y, por lo tanto, $k = (f,g)$.
Concluimos entonces que $G \cap G'$ es el producto de $G$ y $G'$ en $\mathbf{Reg_V}$ pues se satisface la propiedad universal.
Probemos ahora que $G \cup G'$ es el coproducto.
Sea $H$ en $\mathbf{Reg_V}$ junto con $f : G \to H$ y $g : G' \to H$ morfismos de gramáticas regulares. Definimos $[f,g] : G \cup G' \to H$ tal que para cada $v \in (G \cup G')_0$ y $h \in (G \cup G')$ como

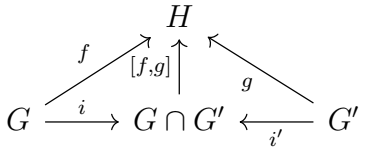
$$[f,g]_0(v) = \begin{cases} f_0(v) & \text{if } v \in G_0 \\ g_0(v) & \text{if } v \in G'_0 \end{cases} \qquad [f,g]_1(h) = \begin{cases} f_1(h) & \text{if } h \in G_1 \\ g_1(h) & \text{if } h \in G'_1 \end{cases}$$

Puesto que $f, g$ son morfismos de gramáticas, el siguiente diagrama conmuta:

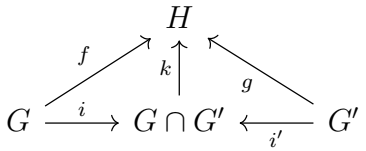



Por la cual tenemos que $L_G(f(h)) = L_H(h) = L_{G'}(g(h))$, y como $[f,g](h) = g(h)$ o $[f,g](h) = g(h)$, entonces $[f,g]$ es un morfismo de gramáticas regulares.

Luego, definimos las inyecciones $i : G \to G \cup G'$ y $i' : G' \to G \cup G'$ como las inmersiones. Es entonces claro que el siguiente diagrama conmuta:




Además, dada cualquier morfismo $k : G \cup G' \to H$ tal que el siguiente diagrama conmute




Entonces para cualesquiera vértice $v$ y arista $h$ en $G$ tenemos que $k_0(v) = k_0 \circ i(v) = f_0(v)$ y $k_1(h) = k_1 \circ i(h) = f_1(h)$, en cambio si están en $G'$ obtenemos que $k_0(v) = k_0 \circ i'(v) = g_0(v)$ y $k_1(h) = k_1 \circ i'(h) = g_1(h)$. Por lo tanto, $[f, h] = k$ como queríamos probar.
Por lo tanto, $G \cup G'$ es el coproducto de $G$ y $G'$. □

# 3 Gramáticas libres de contexto

A continuación introduciremos el concepto de hipercategoría y, siguiendo el camino trazado en el capítulo anterior, veremos los lenguajes que generan estos objetos para luego notar que hemos avanzado en la jerarquía de Chomsky, ya que estos son precisamente los lenguajes libres de contexto, o de tipo 0. Finalmente, implementaremos estos conceptos en `Python`.

## 3.1. Hipercategorías

**Definición 4.** Una **signatura de hipercategoría** es un conjunto de nodos $H_1$ y uno de objetos $H_0$ junto a un par de funciones

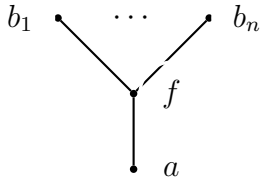
$$H_0^* \xleftarrow{dom} H_1 \xrightarrow{cod} H_0$$

Un morfismo de signaturas de hipercategorías $\varphi : H \to \Gamma$ es un par de funciones $\varphi_0 : H_0 \to \Gamma_0$ y $\varphi_1 : H_1 \to \Gamma_1$ tal que el siguiente diagrama conmuta:

$$\begin{array}{ccccc} H_0^* & \xleftarrow[\texttt{cod}]{} & H_1 & \xrightarrow{\texttt{dom}} & H_1 \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0^* & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_1 \end{array}$$

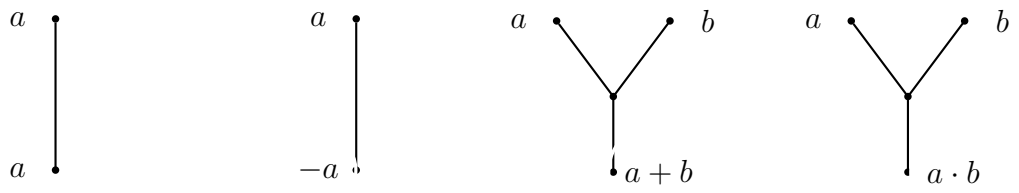
Con tales morfismos, las signaturas de hipergráficas forman la categoría **HypCat**.

Dado una signatura de hipergráfica, denotaremos un nodo $b_1 ... b_n \xrightarrow{f} a$ graficamente como:

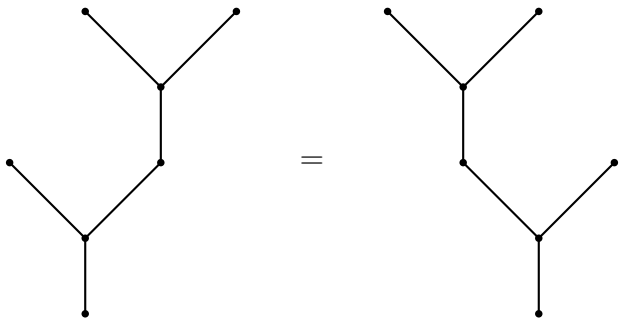



Además, diremos que $f$ tiene aridad $n$.

**Ejemplo 5.** Consideremos la siguiente signatura de hipergráfica donde los objetos corresponderán a los números reales, $\mathbb{R}$. Luego, dado $a, ab \in \mathbb{R}^*$ tenemos los siguientes nodos:




Notemos que los primeros dos nodos corresponden a las operaciones de aridad 1 que lleva un número asímismo o a su inverso aditivo, respectivamente. Por otro lado, los últimos dos nodos corresponden a las operaciones binarias usuales de los números reales: la adición y la multiplicación.
Notemos, además, que para cualquier $abc \in \mathbb{R}^*$ tenemos que, dado que nuestras operaciones son asociativas, entonces $a + (b + c) = (a + b) + c$, así gráficamente tenemos la igualdad de los siguientes diagramas:




Por ello, siguiendo un razonamiento inductivo para cualesquiera $a_1 \dots a_n \in \mathbb{R}$ podemos agregar los nodos:

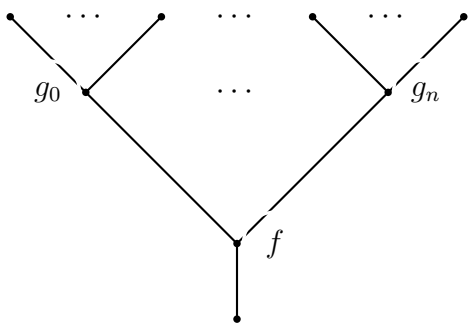

Así, todos los nodos son los representados por alguno de los diagramas anteriores. Con lo anterior, tenemos una signatura de hipercategoría.

Ahora, veamos la definición principal de esta sección:

**Definición 5.** Un **hipercategoría** es una signatura de hipercategoría **H** junto a una operación de composición $\Pi_{b_i \in \vec{b}} \mathbf{Hyp}(\vec{c_i}, b_i) \times \mathbf{Hyp}(\vec{b}, a) \to \mathbf{Hyp}(\vec{c}, a)$ con $a \in \mathbf{O}_0$ y $\vec{c}, \vec{b} \in \mathbf{O}_0^*$. Dadas $f : \vec{b} \to a$ y $g_i : \vec{c_i} \to b_i$ definimos la composición como:

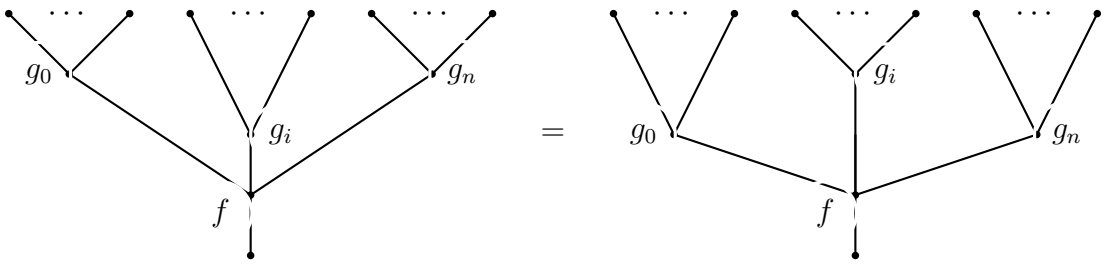
$$(c_1^1, c_2^1, \dots, c_{m_1}^1, c_2^1, \dots, c_1^n, \dots, c_{m_n}^n) \mapsto f(g_1(c_1^1, \dots, c_{m_1}^1), \dots, g_n(c_1^n, \dots, c_{m_n}^n))$$

que representamos gráficamente de la siguiente manera




Además, para cualquier $a \in \mathbf{H}_0$ tenemos una identidad $a \xrightarrow{Id_a} a$ en donde se satisfacen las siguientes propiedades:

1. Para cualquier $f : \vec{b} \to a$ en $\mathbf{O}_1$, $f \cdot Id_a = f = \vec{Id}_{b_i} \cdot f$; y
2. Para cualesquiera $f : \vec{b} \to a$ y $g_i : \vec{c_i} \to b_i$ tenemos que:




Un álgebra $F : \mathbf{H} \to \mathbf{N}$ es un morfismo de signatura de hipercategorías compatible con la composición, es decir, cada que $\vec{c} \xrightarrow{\vec{g}} \vec{b} \xrightarrow{f} a$, $F(\vec{g} \cdot f) = F(\vec{g}) \cdot F(f)$

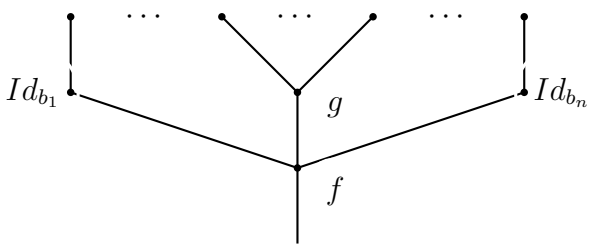
Es importante resaltar que la propiedad 2 nos indica que no importa el orden en que hagamos las composiciones, es decir, se respeta cierta noción de asociatividad. Vale la pena resaltar que todavía no definimos formalmente el significado del cálculo de diagramas presentado, pues lo analizaremos con detenimiento en el siguiente capítulo.
Notemos que la signatura de hipercategoría del ejemplo anterior es propiamente un hipercategoría con la composición dada de la manera obvia.

De manera análoga al capítulo anterior, nuestra intención es definir una hipercategoría libre. Antes, veamos la siguiente definición:

**Definición 6.** Sea $\mathbf{H}$ una hipergráfica. Sea $f : \vec{b} \to a$ y $g : \vec{c} \to b_i$ en $\mathbf{H}_1$ con $i \in \{1, ..., n\}$ con $n$ es la aridad de $f$.
Definimos la composición parcial en la $i-$ésima entrada como:

$$g \circ_i f = (id_{b_1} \cdots g \cdots id_{b_n}) \cdot f$$

Graficamente




Una observación importante es que definimos la composición parcial a partir de la composición, pero también pudimos definirlo inversamente de la siguiente forma:

$$[g_1 \cdots g_n] \cdot f = g_n \circ_n (g_{n-1} \circ_{n-1} \cdots \circ_2 (g_1 \circ_1 f)...)$$

Notemos que la propiedad 2 de la composición para hipergráficas nos garantiza que efectivamente la composición anterior coincide con la usual.
Con ello, ya estamos casi en condiciones de mostrar la construcción de un hipercategoría libre, pero antes veamos una definición más:

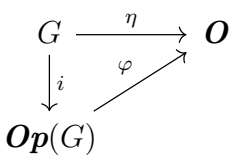
**Definición 7.** Sea $H$ una signatura de hipercategoría, definimos recursivamente un árbol con raíz $a$ de $H$, como:

- Un nodo de la forma $w : \varepsilon \to a$ es un árbol (a los nodos de esta forma, les llamamos hojas);
- Un par $(f, (g_1, \ldots, g_n))$ donde $f$ es un nodo de aridad $n$, $f : a_1 a_2 \cdots a_n \to a$, y $g_1, \ldots, g_n$ son nodos con codominios $a_1, \ldots a_n$ es un arbol;
- Son todos.

Sea $H$ una signatura de operad. $\mathbf{Hyp}(H)$ es la hipercategoría cuyos objetos son exactamente los mismos que $G$ con los morfismos los árboles en $G$ y para cada objeto $a \in G_0$ tenemos la $id_a$. La $i-$ésima composición parcial para morfismos $f, g$ en $\mathbf{Op}(G)$, $g \circ_i f$ está por reemplazar la $i-$ésima hoja de $f$ por una copia de $g$. **La definición puede ser mas rigurosa recursivamente.** Decimos que dos árboles en $\mathbf{Op}(G)$ son iguales si puede ser obtenido uno a partir de otro mediante la aplicación consecutiva de las propiedades 1 y 2 para la composición en operads.
Es claro que la composición parcial anterior induce correctamente una composición de operads y, por lo tanto, $\mathbf{Op}(G)$ efectivamente es un operad.

Más aún, la construcción anterior es libre en virtud de la siguiente proposición:

**Proposición 5.** *Sean $G$ una signatura de operad, $\boldsymbol{O}$ un operad y $f : G_1 \to O_1$ una función que preserve la aridad de los nodos. Entonces existe un único morfismo de operads $\varphi : \boldsymbol{Op}(G) \to \boldsymbol{O}$ tal que el siguiente diagrama conmuta:*



Resta decir que **Op** : **OpSig** $\to$ **Operad** es un funtor y, con ello, tenemos la última proposición de nuestra sección:

**Proposición 6.** *La construcción de un operad libre es parte de la siguiente adjunción:*

$$\boldsymbol{OpSig} \underset{\boldsymbol{U}}{\overset{\boldsymbol{Op}}{\rightleftarrows}} \boldsymbol{Operad}$$

*donde* $\boldsymbol{U}$ *es el funtor olvidadizo de un operad a su signatura.*

## 3.2. Lenguajes libres de contexto

Sea $G$ signatura finita de operad. Consideremos sus objetos como símbolos, los nodos como reglas de producción y los morfismos en **Op**(G) como derivaciones, entonces obtenemos la noción de una gramática, más precisamente:

**Definición 8.** Un gramática operádica como una signatura finita de operad de la siguiente forma:

$$(B + V)^* \leftarrow G \to B$$

donde $V$ es un vocabulario y $B$ es un conjunto de símbolos no terminales con un símbolo distinguido $s \in B$ y $G$ un conjunto de reglas de producciones.
El lenguaje generado por $G$ es:

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \in \mathbf{Op}(G)\}$$

**Agregar dos ejemplo. Lógica proposicional y del lenguaje**

Ahora, como anunciamos al inicio del capítulo, mostraremos la relación entre los lenguajes anteriores con la jerarquía de Chomsky. Para ello, veamos la siguientes definiciones y resultados.

**Definición 9.** Un gramática libre de contexto es una tupla $G = (V, B, P, s)$ donde $V$ es un vocabulario, $B$ es un conjunto de símbolos no terminales con un símbolo distinguido $s$ y $P$ es un conjunto de reglas de producción de la forma $A \to \alpha$ donde $A$ es un símbolo no terminal y $\alpha \in (V + B)^*$.
Si $A \to \alpha$ es una regla de producción, para cualquier $\beta, \gamma \in (V + B)^*$ definimos la derivación $\beta A \gamma \Rightarrow \beta\alpha\gamma$. Consideramos $\Rightarrow^*$ como la cerradura transitiva de $\Rightarrow$.

Definimos el lenguaje generado por $G$ como:

$$\mathcal{L}(G) = \{w \in V^* | \exists s \Rightarrow w\}$$

Decimos que un lenguaje $L$ es libre de contexto si existe una gramática $G$ libre de contexto tal que $L = L(G)$

Observemos que una oración en un lenguaje libre de contexto corresponde a una cadena en $V^*$ que puede ser derivada a partir de $s$ en $G$.

Ahora, la manera anterior de describir el lenguaje es eficiente para la generación de nuevas expresiones, sin embargo, puede no ser muy útil para el propósito de discriminar si una expresión dada pertenece, o no, al lenguaje. Con ese propósito surgen los árboles de derivación.

**Definición 10.** Sea $G = (V, B, P, s)$ una gramática libre de contexto.
Un árbol es un árbol de derivación para $G$ si:

- Cada vértice está etiquetado con un símbolo de $V \cup B \cup \{\varepsilon\}$;
- La raíz del árbol es $s$;
- Si un vértice interior está etiquetado con A, entonces $A \in B$, es decir, no es un símbolo terminal.
- Si un vértice $A$ tiene $n$ hijos, $A_1, \ldots, A_n$, entonces:

  $$A \to A_1 A_2 \cdots A_n$$

  es una regla de producción en $P$.

- Si un vértice está etiquetado con $\varepsilon$, entonces es un hoja y, además, es el único hijo de su padre.

Notemos que el diagrama del ejemplo uno corresponde a un árbol de derivación en la gramática $A \to \neg A | A \wedge A | A \vee A | A \to A | A \leftrightarrow A | a_0 | a_1 | a_2$.
Con esto, podemos enunciar la siguiente proposición cuya prueba podemos consultar en [ Agregar ref].

**Proposición 7.** *Sea $G = (V, B, P, s)$ una gramática libre de contexto. Entonces $s \Rightarrow^* w$ si y sólo si existe un árbol de derivación para $G$ que produce a $w$.*

Es fácil notar que un morfismo en un operad corresponde, de hecho, a un árbol de derivación. Con ello, tenemos el siguiente corolario:

**Teorema 2.** *Sea $G$ una signatura de operads y $G'$ una gramática libre de contexto, entonces*
*1. $L(G)$ es un lenguaje libre de contexto.*
*2. Existe una signatura de operad que genera a $L(G')$.*

*Demostración.* **Pendiente.** □

# 4 Gramáticas recursivamente enumerables

En esta capítulo exploraremos la construcción libre de una categoría monoidal desde una mirada combinatorio con el uso de diagramas, junto a la clase de lenguajes que se pueden generar. Posteriormente, utilizaremos la definición de lenguajes recursivamente enumerables mediante máquinas de Turing para mostrar que estos son exactamente los mismos que los anteriores.

## 4.1. Categorías monoidales

**Definición 11.** Una signatura monoidal $G$ es una signatura de la siguiente forma:

$$G_0^* \xleftarrow{\texttt{dom}} G_1 \xrightarrow{\texttt{cod}} G_0^*$$

$G$ es finita si $G_1$ es finita. Un morfismo de signaturas monoidales $\varphi : G \to \Gamma$ es un par de funciones $\varphi_0 : G_0 \to \Gamma_0$ y $\varphi_1 : G_1 \to \Gamma_1$ tal que el siguiente diagrama conmuta:

$$\begin{array}{ccccc} G_0^* & \xleftarrow[\texttt{cod}]{} & G_1 & \xrightarrow{\texttt{dom}} & G_0^* \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0^* & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_0^* \end{array}$$

Con tales morfismos, las signaturas monoidales forman la categoría **MonSig**.

**Definición 12.** Una categoría monoidal estricta $(\mathcal{C}, \otimes, 1)$ es una categoría $\mathcal{C}$ con un bifuntor[1] $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ y un objeto distinguido $1 \in \mathbf{Ob}(\mathcal{C})$ que satisface las siguientes ecuaciones:

1. $1 \otimes f = f = f \otimes 1$
2. $f \otimes (g \otimes h) = (f \otimes g) \otimes h$

para cualesquiera $f, g, h \in \mathbf{Mor}(\mathcal{C})$
Un funtor monoidal estricto es un funtor entre categorías monoidales $F : \mathcal{C} \to \mathcal{D}$ que preserva el producto tensorial, es decir, $F(g \otimes_{\mathcal{C}} f) = F(g) \otimes_{\mathcal{D}} F(g)$

[1] Un bifuntor es un funtor en cada entrada

Notemos que en una categoría monoidal nuestros morfismos pueden componerse de dos maneras. Dado $f : A \to B$ y $g : B \to C$ morfismos podemos construir la composición $g \circ f : A \to C$. Pero también dados $u : A \to C$ y $v : B \to D$ podemos usar nuestro funtor para formar el morfismo $u \otimes v : A \otimes B \to C \otimes D$.

Es usual interpretar ambas composiciones de la siguiente manera: la composición usual es secuencial, es decir, primero se ejecuta el primer morfismo y después aplicamos el segundo; y la composición funtorial es paralela, es decir, ejecutamos ambos morfismos de manera simultánea.

Las consideraciones anteriores motivan un lenguaje gráfico para categorías monoidales mediante cables y cajas, recursivamente se define de la siguiente manera: Sean $A, B, C, D$ objetos y $f : A \to B$, $g : B \to C$ y $h : C \to D$ morfismos en una categoría monoidal, tenemos la siguiente representación gráfica, llamada diagrama de cables:

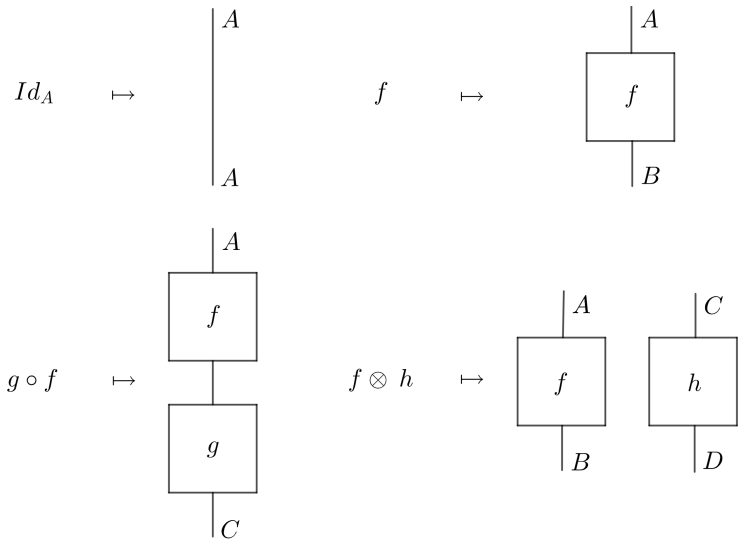



Figura 4.1: Lenguaje gráfico para categorías monoidales

El lenguaje anterior es tan poderoso que Joyal y Street demostraron que dos morfismos en una categoría monoidal son iguales (según los axiomas de categoría monoidal) si y sólo si los diagramas correspondientes son equivalentes. Dos diagramas son equivalentes si uno puede ser deformado, sin cruzar o desconectar cables, en el otro. Lo anterior es conocido como el teorema de coherencia para el cálculo gráfico en categorías monoidales.

Finalmente, es importante señalar que las dos composiciones son compatibles según las leyes de intercambio que enunciaremos a continuación.

**Proposición 8.** *Sea $(\mathcal{C}, \otimes, 1)$ una categoría monoidal y sean $f : A \to B$, $g : B \to C$, $h : D \to E$ y $j : E \to F$ morfismos de $\mathcal{C}$, entonces*

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h) \tag{4.1}$$

$$(Id_B \otimes h) \circ (f \otimes Id_D) = f \otimes h = (f \otimes Id_E) \circ (Id_A \otimes h) \tag{4.2}$$

*Demostración.* Sean $f : A \to B$, $g : B \to C$, $h : D \to E$ y $j : E \to F$ morfismos en una categoría monoidal $\mathcal{C}$.
Primero probemos 4.1.

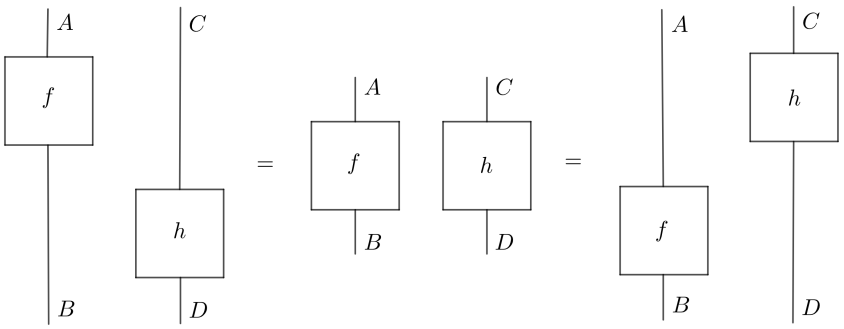
$$\begin{aligned}(g \circ f) \otimes (j \circ h) &= \otimes(g \circ f, j \circ h) \\ &= \otimes((g, j)) \circ (f, h)) \\ &= (\otimes(g, j)) \circ (\otimes(f, h)) \qquad \text{por la bifuntorialidad de } \otimes \\ &= (g \otimes j) \circ (f \otimes h)\end{aligned}$$

Ahora, veamos 4.2

$$\begin{aligned}(Id_B \otimes h) \circ (f \otimes Id_D) &= (Id_B \circ f) \otimes (h \circ Id_D) \qquad \text{Por 4.1} \\ &= f \otimes h \\ &= (f \circ Id_A) \otimes (Id_E \circ h) \\ &= (f \otimes Id_E) \circ (Id_A \otimes h) \qquad \text{Por 4.1}\end{aligned}$$

**Las justificaciónes también podrían escribirse al final, como "donde la tercera igualdad se debe a la bifuntorialidad de $\otimes$"** □

Podemos expresar en el lenguaje gráfico de categorías monoidales la ecuación 4.2 de la siguiente forma:

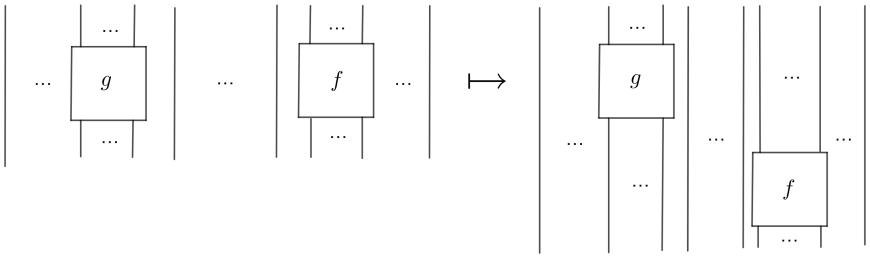



La ecuación anterior sugiere que siempre podemos encontrar diagramas donde cada caja este en un nivel distinto al resto, así que probémoslo, pero antes veamos la siguiente definición para poder enunciar nuestra proposición.

**Definición 13.** Un diagrama de cables está en posición general si a la derecha o izquierda de cada caja no hay otra.

**Proposición 9.** *Todo diagrama de cajas es equivalente a un diagrama de cajas en posición general.*

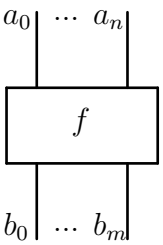
*Demostración.* Sea $D$ un diagrama de cajas. Haremos la demostración por inducción sobre el número de cajas en el diagrama.
La hipótesis de induccion es trivial, pues si $D$ tiene una caja ya acabamos.
Supongamos el resultado es cierto para algún $n$, y que $D$ tiene $n+1$ cajas.
Sea $f$ la caja ubicada más a la derecha y abajo en el diagramas. Consideremos $D'$ el diagrama obtenido al sustituir $f$ por $id_{dom(f)}$. Entonces $D'$ tiene $n$ cajas y por hipótesis es equivalente a un diagrama de cajas $G'$ en posición general, consideremos ahora el diagrama $G$ que se obtiene al pegarle $f$ en el cable al que se lo habíamos cortado. Observemos que (1) la caja que colocamos sigue siendo la más derecha y abajo, y (2) a su izquierda hay a lo más una caja $g$ (si no, ya acabamos), pues $G'$ está en posición general. En ese caso, basta hacer la siguiente deformación:




□

La proposición anterior será una parte fundamental en nuestro próximo objetivo: generar una categoría monoidal libre a partir de una signatura monoidal. Para ello, primero describiremos un lenguaje gráfico para las signaturas monoidales pues lo usaremos en la descripción de categorías monoidales.

Sea $G$ una signatura monoidal y sea $f : \vec{a} \to \vec{b} \in G_1$. Diremos que $f$ es una caja y la representaremos como:




Además, tenemos dos casos distinguidos, cuando $\vec{a}$ o $\vec{b}$ son la palabra vacía, en esos casos diremos que $f$ es un estado o un efecto respectivamente y lo denotaremos

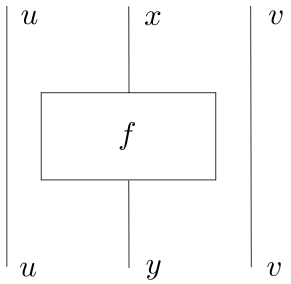
como


Usaremos la descripción anterior para construir la categororía monoidal libre $\mathbf{MC}(G)$ asociada a una signatura monoidal, para ello primero veamos qué es una signatura de niveles.

Dada $G$ una signatura monoidal definimos su signatura de niveles como

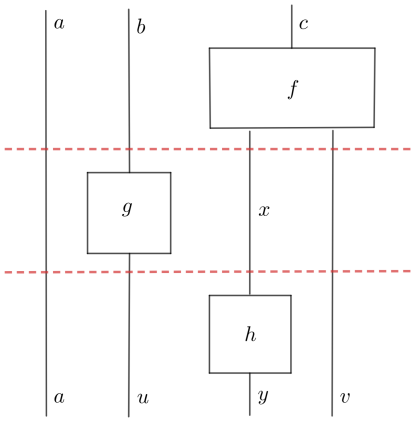
$$G_0^* \xleftarrow[\texttt{cod}]{} L(G) = G_0^* \times G_1 \times G_0^* \xrightarrow{\texttt{dom}} G_0^*$$

donde para cada nivel $l = (u, f : x \to y, v)$ en $L(G)$ definimos $\texttt{dom}(l) = uxv$ y $\texttt{cod}(l) = uyv$, gráficamente representamos a $l$ de la siguiente manera



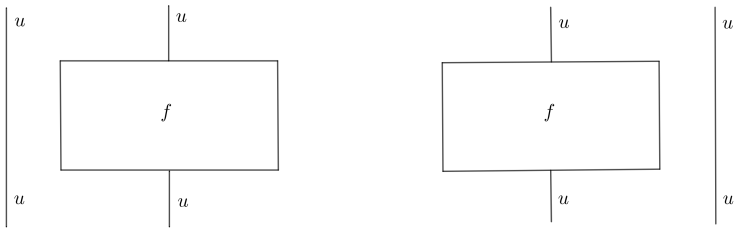

Consideremos ahora $\mathbf{PMC}(G)$, el conjunto de diagramas premonoidales, como el conjunto de morfismos de la categoría libre generada por la signatura de niveles de $G$. Notemos que estos morfismos son secuencias de niveles de la forma $(d_1, \ldots, d_n)$ tales que $\texttt{cod}(d_i) = \texttt{dom}(d_{i+1})$ para cada $i \in \{1, ..., n-1\}$.

Veamos el siguiente ejemplo de un diagrama premonoidal generado con la signatura $G_0 = \{a, b, c, u, v, y, x\}$ y $G_1 = \{f : c \to xv, g : b \to u, h : x \to y\}$



Supongamos que queremos construir el diagrama a partir de la signatura, ¿qué información necesitamos? En principio, conocer el número de niveles que conforma el diagrama, en nuestro caso 3, luego conocer el dominio y codominio del diagrama, *abc* y *auyv* respectivamente, además de la lista de cajas en cada nivel, $[f, g, h]$. Finalmente necesitamos conocer la posición en la que se encuentra la caja en cada nivel, lo cual lo identificamos señalando el número de cables a la izquierda de la caja, $(2, 1, 2)$

Aunque la última condición es innecesaria en el diagrama anterior, es importante en casos como el siguiente, donde los primeros datos no determinan un único diagrama premonoidal.




La explicación anterior motiva nuestra siguiente proposición, una manera eficiente de representar la información contenida en un diagrama premonoidal:

**Proposición 10** (Codificación de diagramas premonoidales)**.** *Sea $G$ una signatura monoidal. Un diagrama premonoidal $d$ en $\boldsymbol{PMC}(G)$ está unicamente determinada de la siguiente forma:*

1. *un dominio:* $\texttt{dom}(d) \in G_0^*$
2. *un codominio:* $\texttt{cod}(d) \in G_0^*$
3. *una lista de cajas:* $\texttt{cajas}(d) \in G_1^n$
4. *una lista de números identificadores:* $\texttt{id}(d) \in \mathbb{N}^n$

*donde $n$ es el número de niveles del diagrama $d$ y el $n-$ésimo identificador señala la posición de la caja en el $n-$ésimo nivel indicando el número de "cables" a la izquierda de la caja.*
*Más aún, la información anterior define un diagrama premonoidal válido si para cada $\imath \in \{1, ..., n\}$ tenemos que*

$$\texttt{peso}(d)_i \geq \texttt{id}(d)_i + |\texttt{dom}(b_i)|$$

*donde*

$$\texttt{peso}(d)_1 = |\texttt{dom}(d)| \qquad \texttt{peso}(d)_{i+1} = \texttt{peso}(d)_i + |\texttt{cod}(b_i)| - |\texttt{dom}(b_i)|$$

*con* $b_i = \texttt{boxes}(d)_i$

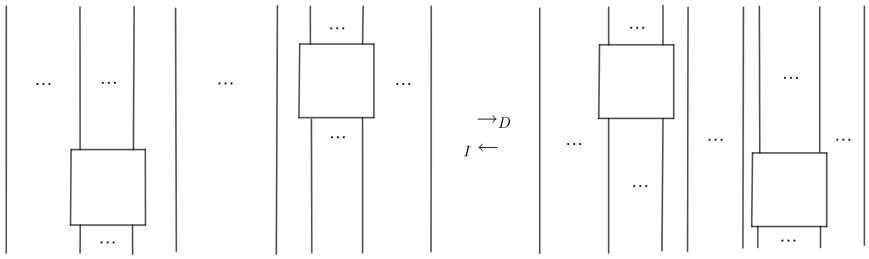
Notemos que la función `peso` indica el número de cables con los que inicia cada nivel, así la desigualdad significa que este número debe ser mayor al número de cables que estarán a la izquierda de la caja menos el número de cables que necesita la caja del nivel. Bajo estas condiciones, es evidente que el diagrama generado será válido.

Ahora bien, notemos que los elementos de $\mathbf{PMC}(G)$ se parecen mucho a diagramas de cables en posición general, sin embargo, necesitamos establecer cuando dos diagramas premonoidales son equivalentes, para ello veamos las siguiente definiciones.

**Definición 14.** Sea $G$ una signatura y $d$ un diagrama premonoidal de $G$. Decimos que $d$ admite un intercambio izquierdo en el nivel $n$ si $\mathtt{id}(d)_{n+1} \geq \mathtt{id}(d)_n + |cod(b_n)|$ y admite un intercambio derecho en el nivel $n+1$ si $\mathtt{id}(d)_n \geq \mathtt{id}(d)_{n+1} + |dom(b_n)|$.

En otras palabras, un intercambio es posible cuando las cajas a intercambiar no tienen cables en común. Ahora sí, veamos que es un intercambio

**Definición 15.** Sea $G$ una signatura y $d$ un diagrama premonoidal de $G$. Si $d$ admite un intercambio izquierdo (o derecho) en el nivel $n$, entonces generamos un nuevo diagrama idéntico en todos los niveles a $d$ salvo en $n$ y $n+1$ donde hay un cambio de la siguiente forma:




decimos que el nuevo diagrama el intercambio izquierdo (o derecho) en el nivel $n$ de $d$.

Claramente los diagramas producidos tras un intercambio es también un diagrama válido. Definimos una reducción de diagramas como una secuencia de intercambios izquierdos o derechos. Con ello, definimos la siquiente relación: dos diagramas $d$ y $d'$ están $\sim$-relacionados si existe una reducción que lleve $d$ a $d'$.

**Proposición 11.** *Sea $G$ una signatura monoidal. La relación $\sim$ sobre $\boldsymbol{PMC}(G)$ es de equivalencia.*

*Demostración.* Sea $G$ una signatura monoidal. Sea $d$ una diagrama premonoidal. Consideramos la secuencia de reducciones vacía, entonces $d \sim d$. Por lo tanto, $\sim$ es reflexiva.
Sea $d$ y $d'$ diagramas tales que $d \sim d'$. Sea $p_1^{r_1}, p_2^{r_2}, ..., p_n^{r_n}$ la secuencia de reducciónes que lleva $d$ a $d'$ con $r_i \in \{d, i\}$ para todo $i\{1, ..., n\}$, es decir, indica si el intercambio fue izquierdo o derecho. Notemos que un intercambio izquierdo es el inverso del derecho y viceversa, entonces consideremos $r_i'$ como el opuesto a $r_i$. Entonces la reducción

$p_n^{r'_n}, p_{n-1}^{r'_{n-1}}, ..., p_1^{r'_1}$ lleva $d'$ a $d$. Por lo tanto, $d' \sim d$. Por lo tanto, $\sim$ es simétrica.
Por último, sean $d$, $d'$ y $d''$ tales que $d \sim d'$ y $d' \sim d''$. Entonces hay una reducción que llevar $d$ a $d'$ y una que lleva $d'$ a $d''$, así hay una reducción que lleva $d$ a $d''$, es decir, $d \sim d''$. Por lo tanto, $\sim$ es transitiva.
Por lo tanto, $\sim$ es de equivalencia. □

Otra forma de expresar la proposición anterior es señalando que cualesquiera dos diagramas premonoidales tales que pueda deformarse, sin cruzar o cortar cables, uno en el otro están $\sim$-relacionados. Con todo lo anterior, ya estamos en condiciones de conocer a nuestra categoría monoidal libre, $\mathbf{MC}(G)$.

**Proposición 12.** *Sea $G$ una signatura monoidal, entonces:*

$$\boldsymbol{MC}(G) \cong \boldsymbol{PMC}(G)/\sim$$

*Demostración.* Sea $G$ una signatura monoidal.
Notemos que cualquier combinación con sentido de elementos de $G_1$ con $\circ$ y $\otimes$ puede expresarse como un diagrama de cables, el cual a su vez puede transformarse a un diagrama equivalente en posición general en virtud de la proposición 9. A su vez, todo diagrama en posición general puede ser expresado como un diagrama premonoidal, por lo tanto, cualquier morfismo en $\mathbf{MC}(G)$ se encuentra en $\mathbf{PMC}(G)$.
Por otro lado, la proposición 11 nos dice que cualquiera dos diagramas premonoidales tales que pueda deformarse, sin cruzar o cortar cables, son iguales. Pero de acuerdo al teorema de coherencia para el cálculo gráfico podemos concluir que $\mathbf{PMC}(G)/\sim$ satisface los axiomas de una categoría monoidal. □

## 4.2. Gramáticas monoidales

Veamos como podemos darle una interpretación lingüística a una signatura monoidal $G$. Podemos considerar los objetos de $\mathbf{MC}(G)$ como cadenas de símbolos de $G_0$, las cajas (o flechas generadores) de $G_1$ como reglas de producción y los morfismos en $\mathbf{MC}(G)$ como derivaciones. Formalmente podemos definir un 'lenguaje monoidal' de la siguiente manera:

**Definición 16.** Una gramática monoidal es una signatura monoidal finita $G$ de la siguiente forma

$$(V+B)^* \xleftarrow{\texttt{dom}} G \xrightarrow{\texttt{cod}} (V+B)^*$$

donde $V$ es un conjunto de palabras llamada vocabulario y $B$ es un conjunto de símbolos con uno inicial $s \in B$, llamado el símbolo de oración.
Una oración $u \in V^*$ es gramaticalmente correcta si hay un morfismos $g : u \to s$ en $\mathbf{MC}$(G). Definimos el lenguaje generado por $G$ como:

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \text{ en } \mathbf{MC}(G)\}$$

A los lenguajes generados por gramáticas monoidales les llamamos lenguajes monoidales.

**Ejemplo 6.** Sabemos que los lenguajes libres de contextos son producidos por gramáticas de hipergráficas de la siguiente forma:

$$(V + B)^* \leftarrow G \rightarrow B$$

Ya que $B \subset (V + B)^*$, entonces son signaturas monoidales y, por lo tanto, es un lenguaje monoidal.

Tal como se prometió al inicio del capítulo, veremos que los lenguajes monoidales son exactamente los lenguajes recursivamente enumerables. Pero antes convendrá recordar las dos definiciones equivalentes, la primera en términos de las gramáticas que los generen y la segunda en términos del modelo de cómputo que los aceptan.

**Definición 17.** Un gramática de tipo 0 (o sin restricciones) es una tupla $G = (V, B, P, s)$ donde $V$ es un vocabulario, $B$ es un conjunto de símbolos no terminales con un símbolo distinguido $s$ y $P$ es un conjunto de reglas de producción de la forma $\alpha \rightarrow \beta$ donde $\alpha, \beta \in (V + B)^*$.
Si $\alpha \rightarrow \beta$ es una regla de producción, para cualquier $\gamma, \delta \in (V + B)^*$ definimos la derivación

$$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$$

Definimos el lenguaje generado por $G$ como:

$$\mathcal{L}(G) = \{v \in V^* | s \Rightarrow^* v\}$$

donde $\Rightarrow^*$ es la cerradura reflexiva y transitiva de $\Rightarrow$.

**Teorema 3.** *Los lenguajes monoidales son exactamente los generados por una gramática de tipo 0.*

*Demostración.* Primero veamos que todo lenguaje monoidal es producido por una gramática de tipo 0.
Sea $G$ una gramática monoidal.
Para cada $f : u \rightarrow v$ en $G_1$ definimos la regla de producción $p_f : v \rightarrow u$ en $P$ con $v, u \in (V + B)^*$. Notemos que si existen dos flechas generadoras $f, g : u \rightarrow v$, entonces $p_f = p_g$.
Sea $w \in \mathcal{L}(G)$, entonces existe $f : w \rightarrow s$ en $\mathbf{MC}(G)$, eso implica existen flechas generadoras tal que el siguiente diagrama conmuta

$$w \xrightarrow{f_1} \cdots \xrightarrow{f_n} s$$
$$\overbrace{\qquad\qquad}^{f}$$

Entonces tenemos la siguiente derivación de $P$:

$$s \Rightarrow \cdots \Rightarrow w$$

Es decir, tenemos que $s \Rightarrow^* w$ y, por lo cual, $w$ está en el lenguaje generado por la grámatica de tipo 0 $(V, B, P, s)$. Por lo tanto, $\mathcal{L}(G) \subset \mathcal{L}(V, B, P, s)$. La otra contención se da leyendo el argumento anterior al revés y, por lo tanto, $\mathcal{L}(G) = \mathcal{L}(V, B, P, s)$. Resta probar que todo lenguaje generado por una gramática de tipo 0 es monoidal. La idea es análoga anterior y, dado que hemos realizado pruebas semejantes en los capítulos anteriores, entonces nos tomaremos la libertad de omitir esta demostración. □

**Ejemplo 7.**

Como se acaba de comentar, la prueba fue sencilla en virtud de la semejanza entre las definiciones de gramática monoidal y gramática de tipo 0. Será más divertido ver como se relaciona nuestra definición categórica con el modelo de computo asociado. Para esto, veamos alguna definiciones preliminares:

**Definición 18.** Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ una máquina de Turing.
Definimos un movimiento de la siguiente manera: Dado $X_1X_2 \cdots X_{i-1}qX_i \cdots X_n$ en la banda y $\delta(q, X_i) = (p, Y, \leftarrow)$ tenemos que:

$$X_1X_2 \cdots X_{i-1}qX_i \cdots X_n \vdash_M X_1X_2 \cdots X_{i-2}pX_{i-1}Y \cdots X_n$$

Pero si $\delta(q, X_i) = (p, Y, \rightarrow)$, entonces

$$X_1X_2 \cdots X_{i-1}qX_i \cdots X_n \vdash_M X_1X_2 \cdots X_{i-1}YpX_{i+1} \cdots X_n$$

Definimos $\vdash_M^*$ como la cerradura reflexiva y transitiva de $\vdash_M$.
El lenguaje $M$ aceptado por una máquina de Turing se define como

$$\mathcal{L}(M) = \{w \in \Sigma^* | q_0w \vdash_M^* \alpha s \beta \text{ con } s \in F \text{ y } \alpha \ , \ \beta \in \Gamma^*\}$$

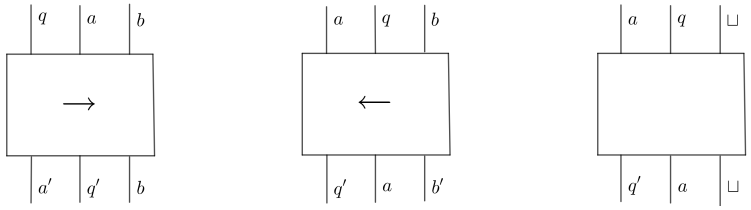
Es decir, el lenguaje producido por una máquina de Turing es el conjunto de cadenas del vocabulario tales que al inicializarse la banda con esas cadenas logramos alcanzar algún estado final. Ya estamos en condiciones de dar la siguiente definición:

**Definición 19.** Un lenguaje $L$ es recursivamente enumerable si existe una máquina de Turing $M$ que acepta $L$, es decir, $L = L(M)$.

Un resultado bien conocido es que que los lenguajes recursivamente enumerables tienen el mismo poder expresivo que los lenguajes generados por una gramática de tipo 0. Veamos, entonces, la manera d

**Proposición 13.** *Para todo lenguaje recursivamente enumerable L, existe una gramática monoidal G tal que* $L = \mathcal{L}(G)$

*Demostración.* Sea $L$ un lenguaje recursivamente enumerable generado por la máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$.
Definimos $B = (\Gamma \setminus \Sigma) + Q$ y $V = \Sigma$ con $s \in B$ el símbolo de oración. Las reglas de producción son las siguientes:




donde $a, a', b, b' \in \Sigma$, $q, q' \in Q$ tales que $\delta(q, a) = (q', a', \rightarrow)$, $\delta(q, b) = (q', b', \leftarrow)$ y $\delta(q, \sqcup) = (q', \sqcup, \rightarrow)$. Además, agregamos las reglas de producción $x \rightarrow q_0 x$ y $xsy \rightarrow s$, donde la segunda sirve para "limpiar la banda".
Veamos que $L = \mathcal{L}(G)$
Sea $w \in L$. Entonces $q_0 w \vdash_M^* \alpha s \beta$. Pero notemos que para cada aplicación $\vdash_M$ tenemos un morfismo que actúa de la misma manera, entonces tenemos morfismos tales que

$$q_0 w \xrightarrow{f_1} ... \xrightarrow{f_n} \alpha s \beta$$

Además como también tenemos las reglas de producción $w \rightarrow q_0 w$ y $\alpha s \beta \rightarrow s$ tenemos una derivación $w \rightarrow s$ en $G$. Por lo tanto, $w \in \mathcal{L}(G)$
Para el regreso, sea $w \in \mathcal{L}(G)$, entonces existe una derivación $w \rightarrow s$.
Observemos que por las reglas producción, tenemos la derivación es necesariamente de la siguiente forma

$$w \longrightarrow q_0 w \longrightarrow ... \longrightarrow \alpha s \beta \longrightarrow s$$

Pero la derivación intermedia solo puede lograrse con reglas de producción de la forma $\leftarrow$ y $\rightarrow$, las cuales corresponden con derivaciones en $\vdash_M$, por lo tanto, $q_0 w \vdash_M^* \alpha w \beta$. Por lo tanto, $w \in L$.
Por lo tanto, $L = \mathcal{L}(G)$ □

Terminemos este capítulo con el siguiente ejemplo

**Ejemplo 8.** El lenguaje $L = \{a^i b^i c^i | i \in \mathbb{N}\}$ no es regular, pero sí es recursivamente enumerable.
Una máquina de Turing que acepte el lenguaje anterior es la siguiente:

| $\delta$ | $a$ | $b$ | $c$ | $x$ | $y$ | $z$ | $\sqcup$ |
|---|---|---|---|---|---|---|---|
| $q_0$ | $q_1, x, \rightarrow$ | | | | $q_4, y, \rightarrow$ | | $q_5, \sqcup, \rightarrow$ |
| $q_1$ | $q_1, a, \rightarrow$ | $q_2, y, \rightarrow$ | | | $q_1, y, \rightarrow$ | | |
| $q_2$ | | $q_2, b, \rightarrow$ | $q_3, z, \leftarrow$ | | | $q_2, z, \rightarrow$ | |
| $q_3$ | $q_3, a, \leftarrow$ | | | $q_0, x, \rightarrow$ | | $q_3, z, \leftarrow$ | |
| $q_4$ | | | | | $q_4, y, \rightarrow$ | $q_4, z, \rightarrow$ | $q_5, \sqcup, \rightarrow$ |
| $q_5$ | | | | | | | $q_5, s, \rightarrow$ |

Veamos la derivación en gramática monoidal correspondiente de la palabra $abc$

# 5 Gramáticas categoriales

**Definición 20.** Una signatura bicerrada $G$ es una colección de generadores $G_1$ y tipos básicos $G_0$ junto a un par de funciones:

$$T(G_0) \xleftarrow{\texttt{dom}} G_1 \xrightarrow{\texttt{cod}} T(G_0)$$

donde $T(G_0)$ es el conjunto de tipos bicerrados dados por la siguiente definición

$$T(G_0) ::== a, b \in G_0 | a \otimes b | a/b | a\backslash b$$

Un morfismo de signaturas bicerradas $\varphi : G \to \Gamma$ es un par de funciones $\varphi_0 : G_0 \to \Gamma_0$ y $\varphi_1 : G_1 \to \Gamma_1$ tales que hacen conmutar el diagrama de signaturas. A la categorías de signaturas monoidales le llamamos **BcSig**.

**Definición 21.** Una categoría monoidal bicerrada $\mathbf{C}$ es una categoría monoidal equipada con dos bifuntores $-\backslash- : \mathbf{C^{op}} \times \mathbf{C} \to \mathbf{C}$ y $-/- : \mathbf{C} \times \mathbf{C^{op}} \to \mathbf{C}$ tales que para cualquier objeto $a$, $a \otimes - \dashv a\backslash-$ y $- \otimes a \dashv -/a$. Es decir, tenemos los siguientes isomorfismos naturales para cualesquiera objetos $a, b, c$

$$\mathbf{C}(a, c/b) \cong \mathbf{C}(a \otimes b, c) \cong \mathbf{C}(b, a\backslash c) \tag{5.1}$$

Estos isos son usualmente llamados curry (cuando $\otimes$ es reemplazado) o uncurry. Con los funtores monoidales como morfismos, las categorías bicerradas forman la categoría **BcCat**.

El desarrollo de las gramáticas que veremos este capítulo estuvo intimamento relacionado con la lógica. Es por esto, que vale la pena mencionar que los morfismos en una categoría bicerrada pueden ser pensados como deducciones en un sistema formal. En particular, los axiomas de una categoría monoidal pueden ser representados como reglas de inferencia de la siguiente manera:

$$\frac{a}{a \to a}\ \texttt{id} \qquad \frac{a \to b \quad b \to c}{a \to c}\ \circ \qquad \frac{a \to b \quad c \to d}{a \otimes c \to c \otimes d}\ \otimes$$

Por su parte, el curry y uncurry en una categoría bicerrada puede ser visto de la siguiente manera:

$$a \to c/b \quad \text{sii} \quad a \otimes b \to c \quad \text{sii} \quad b \to a\backslash c$$

Usualmente pensamos el curry como una manera de transformar una función que toma dos argumentos a una función que toma sólo uno y arroja como resultado otra función.

Dada $G$ una signatura bicerrada, la categoría libre bicerrada $\mathbf{BC}(G)$ contiene todos los morfismos que puede ser generados mediante las cuatro reglas de inferencia anterior por los generadores de $G$.

Ahora bien, veamos la gramática que puede generar este tipo de categorías.

**Definición 22.** Una gramática bicerrada $G$ es un signatura bicerrada de la siguiente forma:

$$T(B+V) \xleftarrow{\texttt{dom}} G \xrightarrow{\texttt{cod}} T(B+V)$$

donde $B$ es un vocabulario y $B$ es un conjunto de tipos básicos con un tipo distinguido $s$. El lenguaje generado por $G$ es

$$\mathcal{L}(G) = \{u \in V^* | \exists f : u \to s \text{ en } \mathbf{BC}(G)\}$$

A continuación veremos que tres tipos de gramáticas importantes, conocidas usualmente como gramáticas categoriales: gramáticas AB, gramáticas de Lambek y gramáticas categoriales combinatorias pueden ser vistas como gramáticas bicerradas. Antes de ello, debemos saber a que nos referimos a encontrar una gramática en otras.

## 5.1. Reducciones gramaticales

**Definición 23.** Sean $G$ y $G'$ gramáticas sobre el mismo vocabulario. Decimos que $G$ se reduce débilmente a $G'$, $G \leq G'$, si $\mathcal{L}(G) \subset \mathcal{L}(G')$. $G$ es débilmente equivalente a $G'$ si $\mathcal{L}(G) = \mathcal{L}(G')$.

Notemos que aunque dos gramáticas sean debilmente equivalente, las derivaciones entre ambas pueden ser completamente distintas. Por ello, necesitamos una definición más fuerte de reducción entre gramáticas.

**Definición 24.** Sean $G$ y $G'$ gramáticas monoidales sobre el mismo vocabulario. Decimos que $G$ se reduce fuertemente a $G'$ si hay un morfismo de signaturas monoidales $\varphi : G \to G'$ tal que $\varphi_0(v) = v$ para todo $v \in V$ y $\varphi_0(s) = s'$. Las gramáticas monoidales con reducciones fuertes como morfismos forman la subcategoría $\mathbf{Grammar}_V$. Dos gramáticas son fuertemente equivalentes si son isomorfas en $\mathbf{Grammar}_V$.

Como es de esperarse, tenemos el siguiente resultado

**Proposición 14.** *Si $G$ se reduce fuertemente a $G'$, entonces $G \leq G'$.*

*Demostración.* Sean $G$ y $G'$ gramáticas monoidales tales que $G$ se reduce fuertemente a $G'$. Veamos que $\mathcal{L}(G) \subset \mathcal{L}(G')$.
Sea $w \in \mathcal{L}(G)$. Entonces existe un morfismo $f : w \to s$ en $\mathbf{MC}(G)$.

Ya que $G$ se reduce fuertemente a $G'$ existe el morfismo de signatura monoidales $\varphi : G \to G'$ tal que $\varphi_0(v) = v$ para todo $v \in V$ y $\varphi_0(s) = s'$. Además, al ser un morfismo de signaturas monoidales el siguiente diagrama conmuta.

$$\begin{array}{ccccc} G_0^* & \xleftarrow[\texttt{cod}]{} & G_1 & \xrightarrow{\texttt{dom}} & G_0^* \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0^* & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_0^* \end{array}$$

Entonces

$$\texttt{dom}(\varphi_1(f)) = \varphi_0(\texttt{dom}(f)) = \varphi_0(w) = w$$

y

$$\texttt{cod}(\varphi_1(f)) = \varphi_0(\texttt{cod}(f)) = \varphi_0(s) = s'$$

Por lo tanto, $\varphi_1(f) : w \to s' \in \mathbf{MC}(G')$. Por lo tanto, $w \in \mathcal{L}(G')$ y así $\mathcal{L}(G) \subset \mathcal{L}(G')$ como queríamos demostrar. □

Notemos que una reducción fuerte es solo re etiquetar las palabras y el tipo de oración, es decir, hacemos esencialmente lo mismo. Así, esta noción puede resultar demasiado fuerte, por ello necesitamos un punto medio entre ambas.

**Definición 25.** Sean $G$ y $G'$ gramáticas monoidales sobre el mismo vocabulario $V$. Una reducción funtorial de $G$ a $G'$ es un funtor $F : \mathbf{MC}(G) \to \mathbf{MC}(G')$ tales que $F_0(v) = v$ para todo $v \in V$ y $F_0(s) = s'$. Una equivalencia funtorial entre $G$ y $G'$ es una reducción funtorial de $G$ a $G'$ y una de $G'$ a $G$.

**Agregar ejemplo para ver que una reducción funtorial es más débil que una fuerte.**

## 5.2. Gramáticas AB

En 1953, Ajdiuciewicz y Bar-Hillel introducen la primera de las gramáticas categoriales, las gramáticas AB. Sea $B$ un conjunto de tipos básicos, definimos los tipos de una gramática AB como

$$T_{AB}(B) ::== a \in B | a\backslash a | a/a$$

A los últimos dos tipos se les conoce como fracciones.
La idea de las gramáticas categoriales, como su nombre lo indica, es asignar a cada palabra una categoría determinada cuyo objetivo es indicar la manera en que las palabras se pueden relacionar entre sí de una manera significativa.

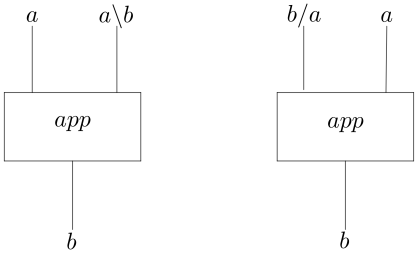
Por ejemplo, consideremos la oración "Valentina come pastel". Notemos que las palabras "Valentina" y "pastel" son sustantivos, así asignemosles el mismo tipo `n`. Por otro lado, la palabra "come" es el núcleo de la oración, así que debería tener el tipo `s`, pero para que tenga sentido a su izquierda necesita tener un sustantivo como

sujeto, el tipo $\mathtt{n}\backslash\mathtt{s}$ representa esto, y a su derecha también tiene otro sustantivo, así asignemosle finalmente el tipo $(\mathtt{n}\backslash\mathtt{s})/\mathtt{n}$. Entonces la derivación correspondiente a la oración "Valentina come pastel" es

$$\mathtt{n} \quad (\mathtt{n}\backslash\mathtt{s})/\mathtt{n} \quad \mathtt{n} \quad \rightarrow \quad \mathtt{n} \quad \mathtt{n}\backslash\mathtt{s} \quad \rightarrow \mathtt{s}$$

Formalmente, definimos una gramática AB de la siguiente manera

**Definición 26.** Una gramática AB es una tupa $G = (V, B, \Delta, s)$ donde $B$ es un vocabulario, $B$ es un conjunto de tipos básicos, $\Delta \subset V \times T_{AB}(B)$ es un léxico. Las reglas de las gramáticas AB están dadas por la siguiente signatura monoidal




con $a, b \in T_{AB}(B)$. El lenguaje generado por $G$ está dado por:

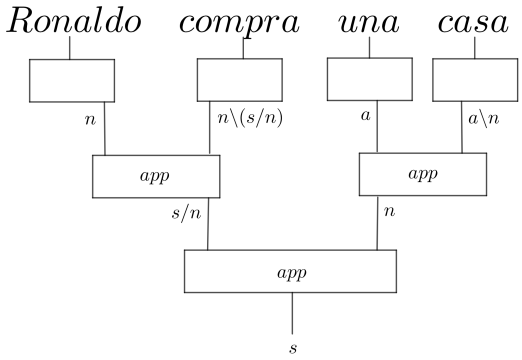
$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \rightarrow s \in \mathbf{MC}(\Delta + R_{AB})\}$$

Veamos el siguiente ejemplo:

**Ejemplo 9.** Consideremos el vocabulario $V = \{Ronaldo, compra, una, casa\}$ y los tipos básicos $B = \{s, n, a\}$ que corresponde a una oración, a un sustantivo y a un artículo. Tomemos el léxico $\Delta$ como

$$\Delta(Ronaldo) = n \quad \Delta(compra) = n\backslash(s/n) \quad \Delta(una) = a \quad \Delta(casa) = a\backslash n$$

Entonces la oración "Ronaldo compra una casa" es gramaticalmente correcta de acuerdo a la siguiente derivación




Veamos el resultado principal de esta sección

**Proposición 15.** *Las gramáticas AB se reducen funtorialmente a una gramática bicerrada.*

*Demostración.* Ya que ambas son categorías monoidales libres, para exhibir un funtor basta de las gramáticas $AB$ a las bicerradas, basta asignar a los generadores $R_{[}AB]$ en la bicerrada, pero veamos que estos se pueden obtener de la definición de categoría bicerrada. Sean $a, b, c$ objetos, entonces

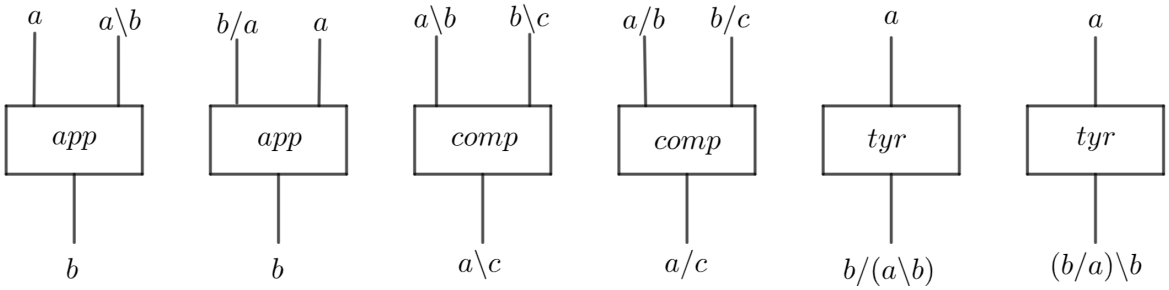
$$\dfrac{\dfrac{a\backslash b}{a\backslash b \to a\backslash b}\,\texttt{id}}{a \otimes (a\backslash b) \to b}\,5.1 \qquad \dfrac{\dfrac{a/b}{a/b \to a/b}\,\texttt{id}}{(a/b) \otimes b \to a}\,5.1$$

como queríamos probar. □

## 5.3. Gramáticas de Lambek

Un par de años más tarde, Joachim Lambek introdujo dos reglas más a las gramáticas categoriales: la composición y el *type raising*. Aunque ambas reglas no le agregan poder expresivo a las gramáticas, sí permite realizar las derivaciones de una manera más literal.

**Definición 27.** Un gramática de Lambek es una tupa $G = (V, B, \Delta, s)$ donde $V$ es un vocabulario, $B$ es un conjunto finito de tipos básicos y $\Delta \subset V \times T(B)$ es un léxico con la siguiente signatura $R_L$




El lenguaje generado por $G$ se define como

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \in \mathbf{MC}(\Delta + R_L)\}$$

**Ejemplo 10.** Pendiente

Notemos que los tipos usados son exactamente los de una categoría bicerrada.

**Proposición 16.** *Las gramáticas AB se reducen funtorialmente a una gramática bicerrada.*

*Demostración.* Como en la proposición anterior, basta ver que podemos generar $R_L$ en una categoría bicerrada. Ya sabemos que las dos *app* es posibles, veamos el resto. Sean $a, b$ objetos. Construyamos las derivaciones correspondientes a *comp*

$$\dfrac{\dfrac{\dfrac{\dfrac{a\backslash c}{a\backslash c \to a\backslash c}\,\mathtt{id}}{a \otimes a\backslash b \to b}\,5.1 \qquad \dfrac{b\backslash c}{b\backslash c \to b\backslash c}\,\mathtt{id}}{a \otimes a\backslash b \otimes b\backslash c \to c}\,\otimes \text{ y } \circ}{a\backslash b \otimes b\backslash c \to a\backslash c}\,5.1$$

$$\dfrac{\dfrac{a/b}{a/b \to a/b}\,\mathtt{id} \qquad \dfrac{\dfrac{b/c}{b/c \to b/c}\,\mathtt{id}}{b/c \otimes c \to b}\,5.1}{a/b \otimes b/c \otimes c \to a}\,\otimes \text{ y } \circ}{a/b \otimes b/c \to a/c}\,5.1$$

Concluyamos con las derivaciones de *tyr*.

$$\dfrac{\dfrac{\dfrac{a\backslash b}{a\backslash b \to a\backslash b}\,\mathtt{id}}{a \otimes (a\backslash b) \to b}\,5.1}{a \to b/(a\backslash b)}\,5.1$$

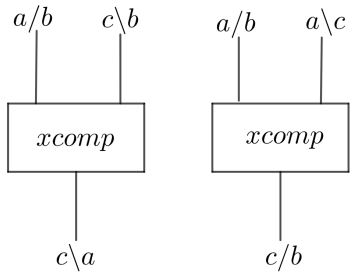
$$\dfrac{\dfrac{\dfrac{b/a}{b/a \to b/a}\,\mathtt{id}}{(b/a) \otimes a \to b}\,5.1}{a \to (b/a)\backslash b}\,5.1$$

con lo cual terminamos la prueba. □

## 5.4. Gramáticas combinatoria

En los ochentas, Stuart Schieber demostró que hay lenguas que no pueden ser generadas por gramáticas libre de contexto, essto es porque ciertas oraciones involucran de manera natural dependencias cruzadas, fenómeno que ocurre cuando dos series de palabras se mezclan entre sí. AUnque el fenomeno no es usual en español o inglés, es muy común en el suizo-alemán, veamos el siguiente ejemplo:
... mer em Hans es huss halfed aastriiche
Que se traduce como: ... nosotros ayudamos a Hans a pintar la casa.
Otro ejemplo sencillo es la oración: "Él considera incompetente a cualquiera que...".
Para analizar estos caso, se extienden la gramáticas anteriores a la gramática categorial combinatoria mediante la inclusión de dos reglas que composición cruzada.

**Definición 28.** Una GCC es una tupla $G = (V, B, \Delta, s)$ donde $V$ es un vocabulario, $B$ es un conjuntos de tipos y $\Delta \subset V \times T(B)$ es un léxico. La signatura $R_{GCC}$ es la signatura $R_L$ más las siguientes dos reglas para todo $a, b, c$ objetos:



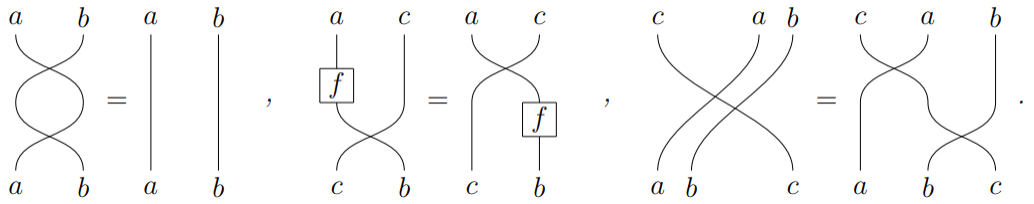

El lenguaje generado por $G$ se define como

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \in \mathbf{MC}(\Delta + R_{GCC})\}$$

**Ejemplo 11.** Pendiente

A diferencia de las dos gramáticas categoriales anteriores, la signatura de *CCG* no puede ser deducida de los axiomas de categoría bicerrada, pues todos los axiomas preservan el orden de los tipos. Así, nuestro mejor intentó sería agregar directamente las reglas *xcomp* a la signatura de las gramáticas monoidales, o bien, trabajar en categorías cerradas.

**Definición 29.** Una categoría monoidal simétrica $\mathbf{C}$ es una categoría monoidal junto a una transformación natural $\sigma : a \otimes b \to b \otimes a$ que satisface




para cualesquiera $a, b, c \in textbfC_0$ y $f : a \to b \in \mathbf{C}_1$.

**Definición 30.** Una categoría cerrada es una categoría simétrica bicerrada.

**Proposición 17.** *Una categoría cerrada tiene las reglas xcomp como morfismos.*

*Demostración.* Sean $a, b, c$ objetos una categoría cerrada libre. Basta mostrar las derivaciones para *xcomp*.

$$\dfrac{\dfrac{a/b}{a/b \to a/b}\,\texttt{id} \qquad \dfrac{\dfrac{c\backslash b}{c\backslash b \to c\backslash b}\,\texttt{id}}{c \otimes c\backslash b \to b}\,5.1}{\dfrac{a/b \otimes c \otimes c\backslash b \to a}{c \otimes a/b \otimes c\backslash b \to a}\,\sigma}{a/b \otimes c\backslash b \to c\backslash a}\,5.1 \qquad \otimes \text{ y } \circ$$

$$\dfrac{\dfrac{\dfrac{a/b}{a/b \to a/b}\,\texttt{id}}{a/b \otimes b \to a}\,5.1 \qquad \dfrac{a\backslash c}{a\backslash c \to a\backslash c}\,\texttt{id}}{\dfrac{a/b \otimes b \otimes a\backslash c \to c}{a/b \otimes a\backslash c \otimes b \to c}\,\sigma}{a/b \otimes a\backslash c \to c/b}\,5.1 \qquad \otimes \text{ y } \circ$$

□

Una última observación es que podría resultar una mala idea trabajar gramáticas sobre categorías cerradas sin restricciones, pues entonces corremos el riesgo de perder el orden de las palabras en una oración. Sin embargo, con la aplicación exclusiva de *xcomp* lo anterior se sospecha no ocurre.

# 6 Pregrupos

hola